```
# -*- coding: utf-8 -*-
"""
Example of how to use ObsPy to tune STA/LTA parameters for the Pav
quakes recorded on the joint seismic+infrasound station PN7A on 20
and extract event start and end times, and corresponding traces. A
step would allow these traces to be imported into GISMO as wavefor
objects.

Created on Sat Apr 23 10:53:12 2016

@author: glennthompson
"""

# Load 1 day of data, trim to 5 minutes, plot time series & spectr
import sys
sys.path.append('/Users/glennthompson/Dropbox/scratch_matlab')
import tune_sta_lta as tsl
from obspy.core import read
from obspy.core.utcdatetime import UTCDateTime
import obspy.signal.trigger as trigger
tstart = UTCDateTime(2007, 8, 28, 3, 55, 0)
tend = UTCDateTime(2007, 8, 28, 4, 0, 0)
st = read("/Users/glennthompson/Dropbox/scratch_matlab/SEEDDATA/PN
st.plot()
st.spectrogram()

# Filter from 2-10 Hz
st.filter('bandpass', freqmin=2.0, freqmax=10.0, corners=2, zeroph
st.plot(type='relative', equal_scale=False)

# Downsample from 100 Hz to 50 Hz (anti-aliasing filter applied)
# This is just so that STA/LTA runs faster
# st.decimate(factor=2, strict_length=False)

# This is the main signal we are trying to capture - and the time
# will maximise the STA/LTA ratio in
t_signal_start = 130.0
t_signal_end = 150.0
st.plot(type='relative', equal_scale=False, starttime=tstart+t_sig

# Run the STA/LTA tuner on the infrasound trace to find the best w
algorithm = 'classic_sta_lta'
numtries = 100
tr_infrasound = st[0]
tr_seismic = st[1]
result = tsl.tune_sta_lta(tr_infrasound, algorithm, t_signal_start
```

```
sta_best = result[0]
lta_best = result[1]
print "Best STA window = %.1f seconds, Best LTA window = %.1f seco

# Show the STA/LTA ratio
staltaratio_best = result[2]
thresh_on = 5
thresh_off = 2.5
trigger.plot_trigger(tr_infrasound, staltaratio_best, thresh_on, t

# Create trigger events by applying these same best STA/LTA settin
triggers_per_event = 1 # set this to 2 and it would have to trigge
import re # for some dumb reason, coincidence trigger needs algori
algorithm_without_underlines = re.sub('_', '', algorithm)
trig = trigger.coincidence_trigger(algorithm_without_underlines, t
from pprint import pprint
pprint(trig)
print "Number of events detected = %d" % len(trig)

# Plot each trigger
pretrig = 5;
posttrig = 5;
count = 0
for thistrig in trig:
    count += 1
    print "Event %d at %s" % (count,thistrig['time'].isoformat())
    st2 = st.copy()
    st2.trim(starttime = thistrig['time'] - pretrig, endtime = thi
    st2.plot(type='relative', equal_scale=False)
```